# State of the Art Cocoa

## Application Development

## Session 422

These are confidential sessions—please refrain from streaming, blogging, or taking pictures.

# State of the Art Cocoa

## Application Development

**Jon Hess**
**Mike Ferris**

# Today's Topics

- Code Clarity
- Maximum Performance
- State of the Art Media
- Dealing with Data
- Plays Well with Others
- An App for Everyone

# Code Clarity

# Simplify Object Lifecycle Management

- Garbage Collection
- Properties

# Garbage Collection

## Less code, better performance

- Eliminates leaks and cycles
- Threaded collection performance
- CFTypes are opt-in

# Properties

## Before properties

```
// MyPerson.h
@interface MyPerson : NSObject {...}
- (NSString *)fullName;
- (NSString *)firstName;
- (void)setFirstName:(NSString *)value;
- (NSString *)lastName;
- (void)setLastName:(NSString *)value;
@end
```

# Properties

## Before properties

```
// MyPerson.m
@implementation MyPerson
- (NSString *)firstName {
    return [[firstName retain] autorelease];
}

- (void)setFirstName:(NSString *)value {
    if (firstName != value) {
        [firstName release];
        firstName = [value copy];
    }
}
```

# Properties

## After properties

```
// MyPerson.h
@interface MyPerson : NSObject {...}
@property (copy) NSString *firstName;
@property (copy) NSString *lastName;
@property (readonly) NSString *fullName;
@end
```

# Properties

## After properties

```
// MyPerson.h
@interface MyPerson : NSObject {...}
@property (copy) NSString *firstName;
@property (copy) NSString *lastName;
@property (readonly) NSString *fullName;
@end
```

```
// MyPerson.m
@implementation MyPerson
@synthesize firstName;
@synthesize lastName;
- (NSString *)fullName {
    return ...;
}
@end
```

# Properties

## Dot syntax

```
MyPerson *person = ...
person.firstName = @“Bill”;
person.lastName = @“Smith”;
NSLog(@“%@”, person.fullName);
NSLog(@“%@”, [person fullName]);
```

# Properties

## Clarify intent

- More concise expression of API
- Clearly differentiates between state and behavior
- More precise code completion

# More About Garbage Collection and Properties

| Objective-C and Garbage Collection Advancements | Russian Hill<br>Wednesday, 5:00PM |
| --- | --- |

## Documentation

Garbage Collection Programming Guide

# Eliminate MVC Glue

- Key-value observation
- Cocoa bindings

# Key-Value Observation

- Eliminates coupling between model and controllers
- Clearly separates concern
- Built on key-value coding
- Foundation for Cocoa bindings

- Allows clients to pick what they observe, and how
- Make your model objects observable

# Cocoa Bindings

- Automates data flow between model and view
- Focuses controllers on more interesting behavior
- Great multiple, and empty selection support
- Leverage NSController subclasses
- Integrated into Interface Builder

# More About Cocoa Bindings

| Exploring Cocoa Bindings Tutorial Lab | OS X Lab B<br>Tuesday, 3:30PM |
| --- | --- |

## Documentation

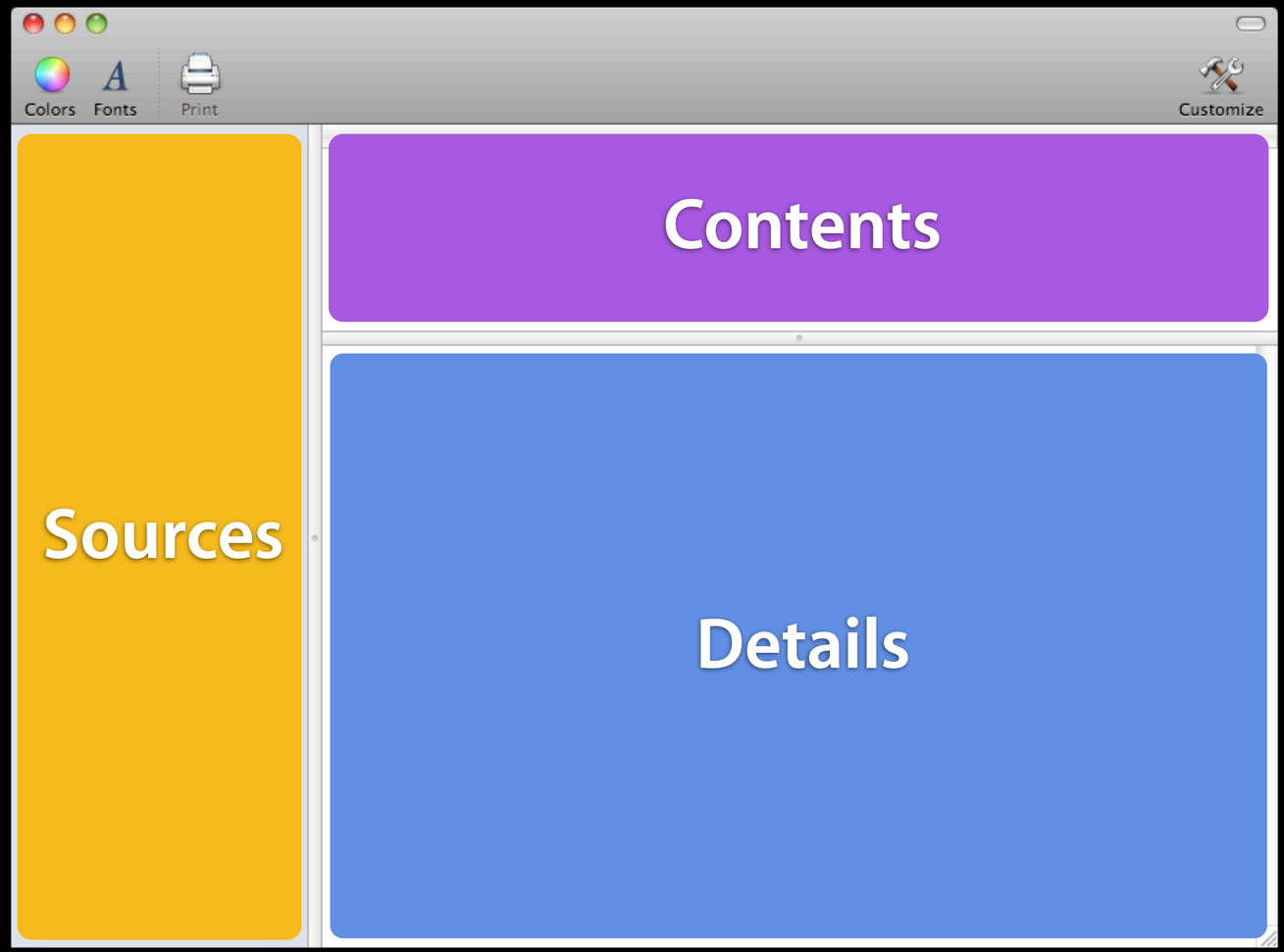
Cocoa Bindings Programming Topics

# Refactoring Opportunities

- View controllers
- Blocks

# View Controllers

## Divide responsibilities



# View Controllers

## Refactor your UI

- Factor multicomponent windows
- One class per area of responsibility
- Improves code focus and reuse
- NSViewController, and UIViewController

# Blocks

- Eliminates callback context management
- New method of encapsulation
- Reduces indirection without increasing coupling
- Common execution unit in GCD

- Blocks are objects

# Blocks

## An example

```
- (NSArray *)stablySortPeople:(NSArray *)people byKey:(NSString *)key {
    NSMapTable *order = [NSMapTable mapTableWithStrongToStrongObjects];
    NSInteger idx = 0;
    for(Person *person in people) {
        [order setObject:[NSNumber numberWithInteger:idx] forKey:person];
        idx += 1;
    }
    MyContext context = {.key = key, .order = order};
    return [people sortedArrayUsingFunction:MyComparePeople context:&context];
}
```

# Blocks

## An example

```
- (NSArray *)stablySortPeople:(NSArray *)people byKey:(NSString *)key {
    NSMapTable *order = [NSMapTable mapTableWithStrongToStrongObjects];
    NSInteger idx = 0;
    for(Person *person in people) {
        [order setObject:[NSNumber numberWithInteger:idx] forKey:person];
        idx += 1;
    }
    MyContext context = {.key = key, .order = order};
    return [people sortedArrayUsingFunction:MyComparePeople context:&context];
}
```

# Blocks

## An example

```
- (NSArray *)stablySortPeople:(NSArray *)people byKey:(NSString *)key {
    NSMapTable *order = [NSMapTable mapTableWithStrongToStrongObjects];
    NSInteger idx = 0;
    for(Person *person in people) {
        [order setObject:[NSNumber numberWithInteger:idx] forKey:person];
        idx += 1;
    }
    MyContext context = {.key = key, .order = order};
    return [people sortedArrayUsingFunction:MyComparePeople context:&context];
}
```

# Blocks

## An example

```
- (NSArray *)stablySortPeople:(NSArray *)people byKey:(NSString *)key {
    NSMapTable *order = [NSMapTable mapTableWithStrongToStrongObjects];
    NSInteger idx = 0;
    for(Person *person in people) {
        [order setObject:[NSNumber numberWithInteger:idx] forKey:person];
        idx += 1;
    }
    MyContext context = {.key = key, .order = order};
    return [people sortedArrayUsingFunction:MyComparePeople context:&context];
}
```

# Blocks

## An example

```
- (NSArray *)stablySortPeople:(NSArray *)people byKey:(NSString *)key {
    NSMapTable *order = [NSMapTable mapTableWithStrongToStrongObjects];
    NSInteger idx = 0;
    for(Person *person in people) {
        [order setObject:[NSNumber numberWithInteger:idx] forKey:person];
        idx += 1;
    }
    MyContext context = {.key = key, .order = order};
    return [people sortedArrayUsingFunction:MyComparePeople context:&context];
}
```

# Blocks

## An example

```
typedef struct {
    NSString *key;
    NSMapTable *order;
} MyContext;

NSComparisonResult MyComparePeople(id personA, id personB, void *context) {
    MyContext *myContext = context;
    id valueA = [personA valueForKey:myContext->key];
    id valueB = [personB valueForKey:myContext->key];
    NSNumber *positionA = [myContext->order objectForKey:personA];
    NSNumber *positionB = [myContext->order objectForKey:personB];
    NSComparisonResult valueOrder = [valueA compare:valueB];
    return valueOrder ? valueOrder : [positionA compare:positionB];
}
```

# Blocks

## An example

```
typedef struct {
    NSString *key;
    NSMapTable *order;
} MyContext;

NSComparisonResult MyComparePeople(id personA, id personB, void *context) {
    MyContext *myContext = context;
    id valueA = [personA valueForKey:myContext->key];
    id valueB = [personB valueForKey:myContext->key];
    NSNumber *positionA = [myContext->order objectForKey:personA];
    NSNumber *positionB = [myContext->order objectForKey:personB];
    NSComparisonResult valueOrder = [valueA compare:valueB];
    return valueOrder ? valueOrder : [positionA compare:positionB];
}
```

# Blocks

## An example

```
- (NSArray *)stablySortPeople:(NSArray *)people byKey:(NSString *)key {
    NSMapTable *order = [NSMapTable mapTableWithStrongToStrongObjects];
    [people enumerateObjectsUsingBlock:^(id person, NSUInteger idx, BOOL *stop) {
        [order setObject:[NSNumber numberWithInteger:idx] forKey:person];
    }];
    return [people sortedArrayUsingComparator:^(id personA, id personB) {
        id valueA = [personA valueForKey:key];
        id valueB = [personB valueForKey:key];
        NSNumber *positionA = [order objectForKey:personA];
        NSNumber *positionB = [order objectForKey:personB];
        NSComparisonResult valueOrder = [valueA compare:valueB];
        return valueOrder ? valueOrder : [positionA compare:positionB];
    }];
}
```

# Blocks

## An example

```
- (NSArray *)stablySortPeople:(NSArray *)people byKey:(NSString *)key {
    NSMapTable *order = [NSMapTable mapTableWithStrongToStrongObjects];
    [people enumerateObjectsUsingBlock:^(id person, NSUInteger idx, BOOL *stop) {
        [order setObject:[NSNumber numberWithInteger:idx] forKey:person];
    }];
    return [people sortedArrayUsingComparator:^(id personA, id personB) {
        id valueA = [personA valueForKey:key];
        id valueB = [personB valueForKey:key];
        NSNumber *positionA = [order objectForKey:personA];
        NSNumber *positionB = [order objectForKey:personB];
        NSComparisonResult valueOrder = [valueA compare:valueB];
        return valueOrder ? valueOrder : [positionA compare:positionB];
    }];
}
```

# Blocks

## An example

```
- (NSArray *)stablySortPeople:(NSArray *)people byKey:(NSString *)key {
    NSMapTable *order = [NSMapTable mapTableWithStrongToStrongObjects];
    [people enumerateObjectsUsingBlock:^(id person, NSUInteger idx, BOOL *stop) {
        [order setObject:[NSNumber numberWithInteger:idx] forKey:person];
    }];
    return [people sortedArrayUsingComparator:^(id personA, id personB) {
        id valueA = [personA valueForKey:key];
        id valueB = [personB valueForKey:key];
        NSNumber *positionA = [order objectForKey:personA];
        NSNumber *positionB = [order objectForKey:personB];
        NSComparisonResult valueOrder = [valueA compare:valueB];
        return valueOrder ? valueOrder : [positionA compare:positionB];
    }];
}
```

# Blocks

## An example

- Eliminates out of band struct
- Allows coding in conceptual order
- Keeps file namespace tidy

# More About Blocks

| | |
|---|---|
| Programming with Blocks and Grand Central Dispatch | Marina<br>Wednesday, 2:00PM |

# Maximum Performance

# 64-Bit

- Faster code for x86-64
- New Objective-C runtime
  - No fragile base class
  - Instance variable synthesis
  - Unified zero cost exceptions
- All apps should be 64-bit in 10.6
  - *Don't* be the last app to pull in 32-bit libraries!
- Use the 64-bit transition guide

# Grand Central Dispatch

- Automatically scaling asynchronous dispatch
  - Thread pool with just the right number of threads
- Very convenient to schedule
- Much more memory efficient than straight threads
- New synchronization concepts
- Much less code
- Event monitoring

# NSOperation and NSOperationQueue

- Easily schedule asynchronous operations
- Dependency management
- Cocoa abstraction to GCD
- Available on Leopard

# More About GCD and NSOperation

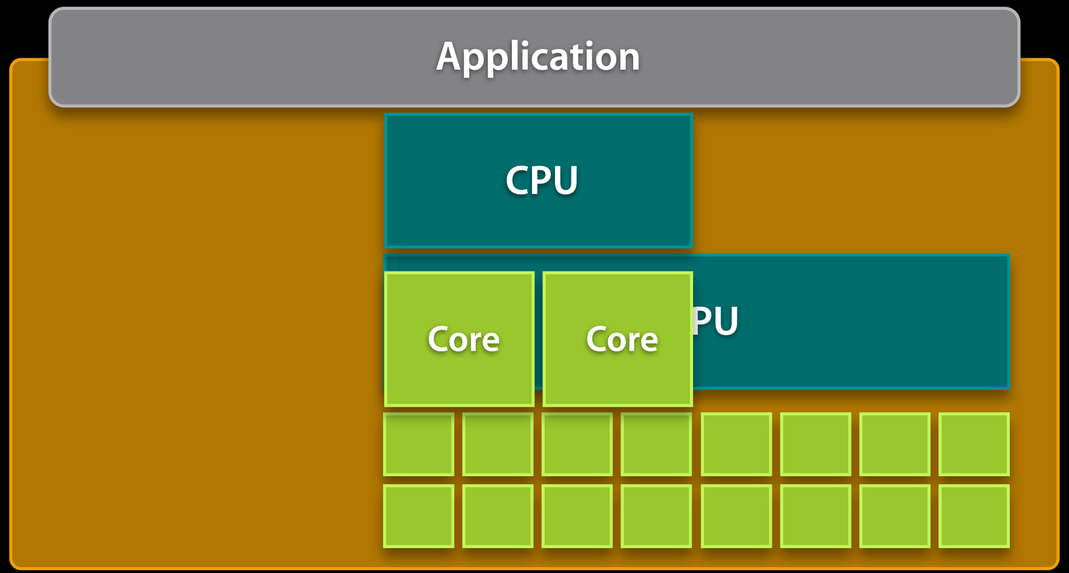
| | |
|---|---|
| Concurrent Programming in Cocoa | North Beach<br>Wednesday, 9:00AM |
| Designing Your Cocoa Application for Concurrency | Nob Hill<br>Tuesday, 3:30PM |
| Programming with Blocks and Grand Central Dispatch | Marina<br>Wednesday, 2:00PM |
| Migrating Your Application to Grand Central Dispatch | Pacific Heights<br>Thursday, 2:00PM |
| Understanding Grand Central Dispatch in Depth | Presidio<br>Thursday, 9:00AM |

## Documentation

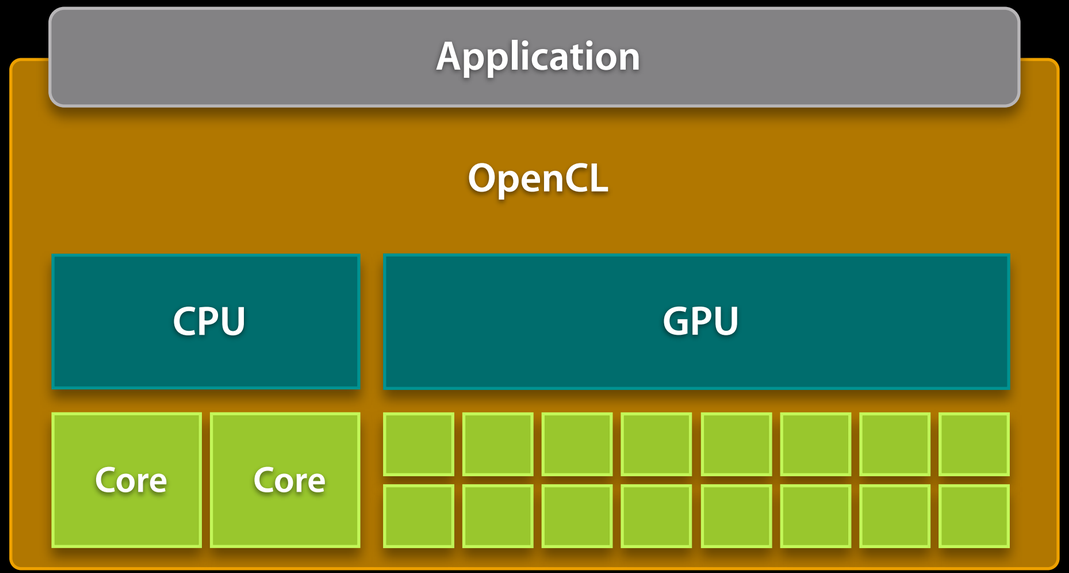
Concurrency Programming Guide

CPU

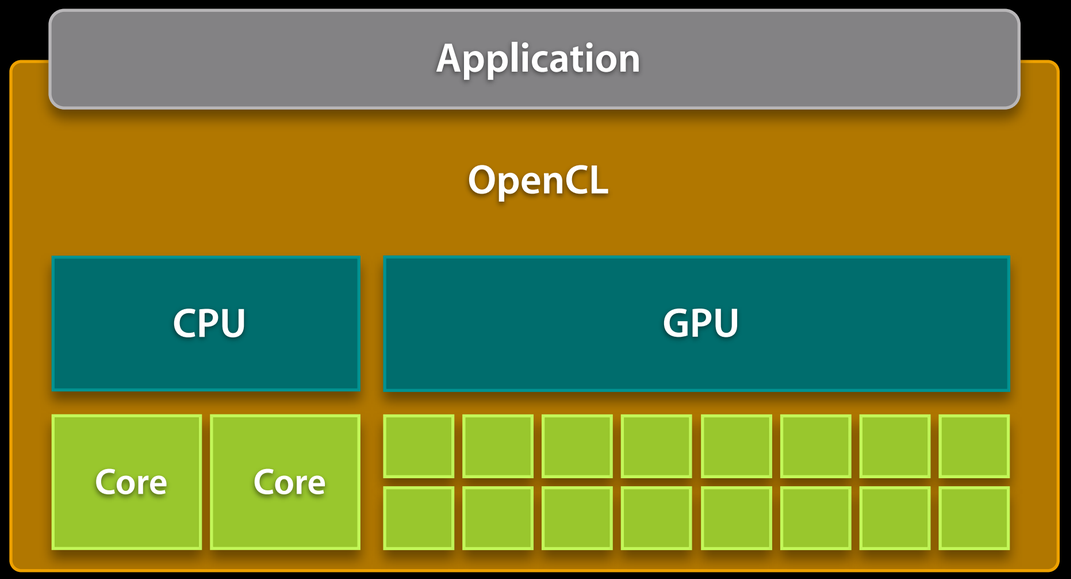
OpenCL


Application
CPU
Core
Core
PU

OpenCL


Application
OpenCL
CPU
GPU
Core
Core

# OpenCL




- Many available execution units
- Great for data-parallel tasks
- GPU is 2x–10x faster than CPU
- Cross-platform open standard
- Uses familiar C99-like language

Shadows on some things, not all?

# More About OpenCL

| | |
|---|---|
| Introduction to OpenCL | Marina<br>Thursday, 9:00AM |
| Harnessing the Power of OpenCL | Marina<br>Thursday, 10:30AM |
| Mastering OpenCL | Marina<br>Thursday, 2:00PM |
| OpenCL Lab | Graphics and Media Lab B<br>Thursday, 3:30PM |

# State of the Art Media

# QT Kit

- Playback, editing, capture, import, and export
- Powered by QuickTime X
  - Lightweight media playback
  - Improved thread safety
  - Fully 64-bit
  - Supports modern codecs

# More About QuickTime

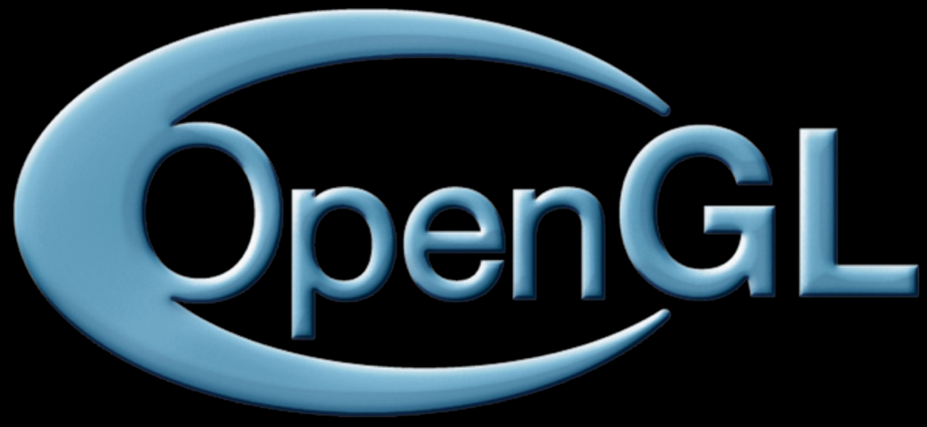
| | |
|---|---|
| Transitioning to QuickTime X | Marina<br>Tuesday, 2:00PM |
| QT Kit Lab | Graphics and Media Lab B<br>Tuesday, 3:30PM |

## Documentation

QuickTime Kit Programming Guide

# OpenGL

- Fast, low-level, portable 2D and 3D graphics
- Hardware accelerated
- Primary interface to the GPU
- Open GL 3.0 is improving rapidly



# More About OpenGL

| | |
|---|---|
| OpenGL Techniques for Snow Leopard | Marina<br>Wednesday, 10:30AM |
| OpenGL for the Mac Lab | Graphics and Media Lab B<br>Thursday, 12:00PM |

# Core Animation

- Familiar object abstraction for animation
- Leverages GPU automatically
- Implicit and explicit animations
- Integrate seamlessly with other technologies
  - QuickTime
  - OpenGL
  - Quartz Composer
  - Cocoa and Cocoa Touch

# More About Core Animation

| | |
|---|---|
| Core Animation Techniques for iPhone and Mac | Mission<br>Wednesday, 9:00AM |
| Core Animation Lab | Graphics and Media Lab B<br>Wednesday, 12:00PM |

## Documentation

Core Animation Programming Guide

# Quartz Composer

# Quartz Composer

- Visually build data-driven effects and transitions
- Immediate visual feedback during design
- Bridges many graphics technologies
- Integrated with Cocoa and Interface Builder

# More About Quartz Composer

| | |
|---|---|
| Quartz Composer Advances in Snow Leopard | Marina<br>Wednesday, 5:00PM |
| Quartz Composer Lab | Graphics and Media Lab A<br>Thursday, 3:30PM |

## Documentation

Quartz Composer Programming Guide

# Gestures and Multi-Touch

- Direct manipulation—improved UI
- Leverage Multi-Touch on Mac OS X
- All current laptops support Multi-Touch

# More About Gestures and Multi-Touch

| | |
|---|---|
| User Events in Cocoa | Nob Hill<br>Wednesday, 3:30PM |

# Core Location

- Create location-based services
- Live updates of user location
- CoreLocation.framework with CLLocationManager

# More About Core Location

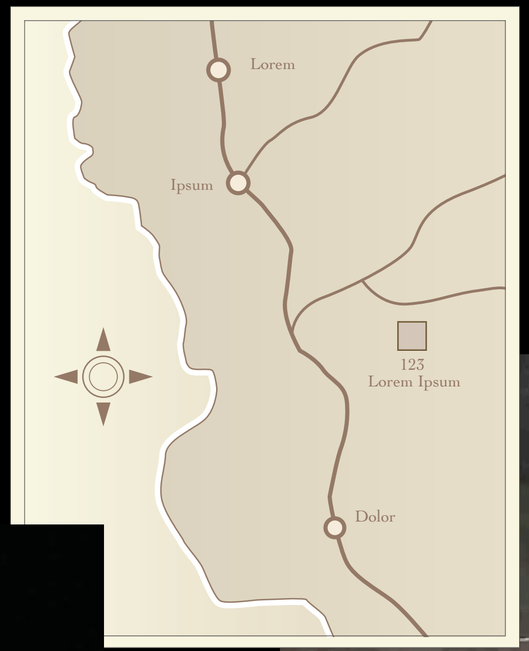
| | |
|---|---|
| Finding Your Place with Core Location | Presidio<br>Thursday, 2:00PM |

## Documentation

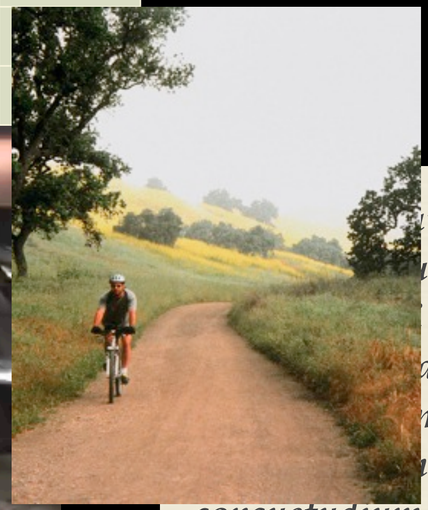
Core Location Framework Reference

# Dealing with Data

# Got Data?



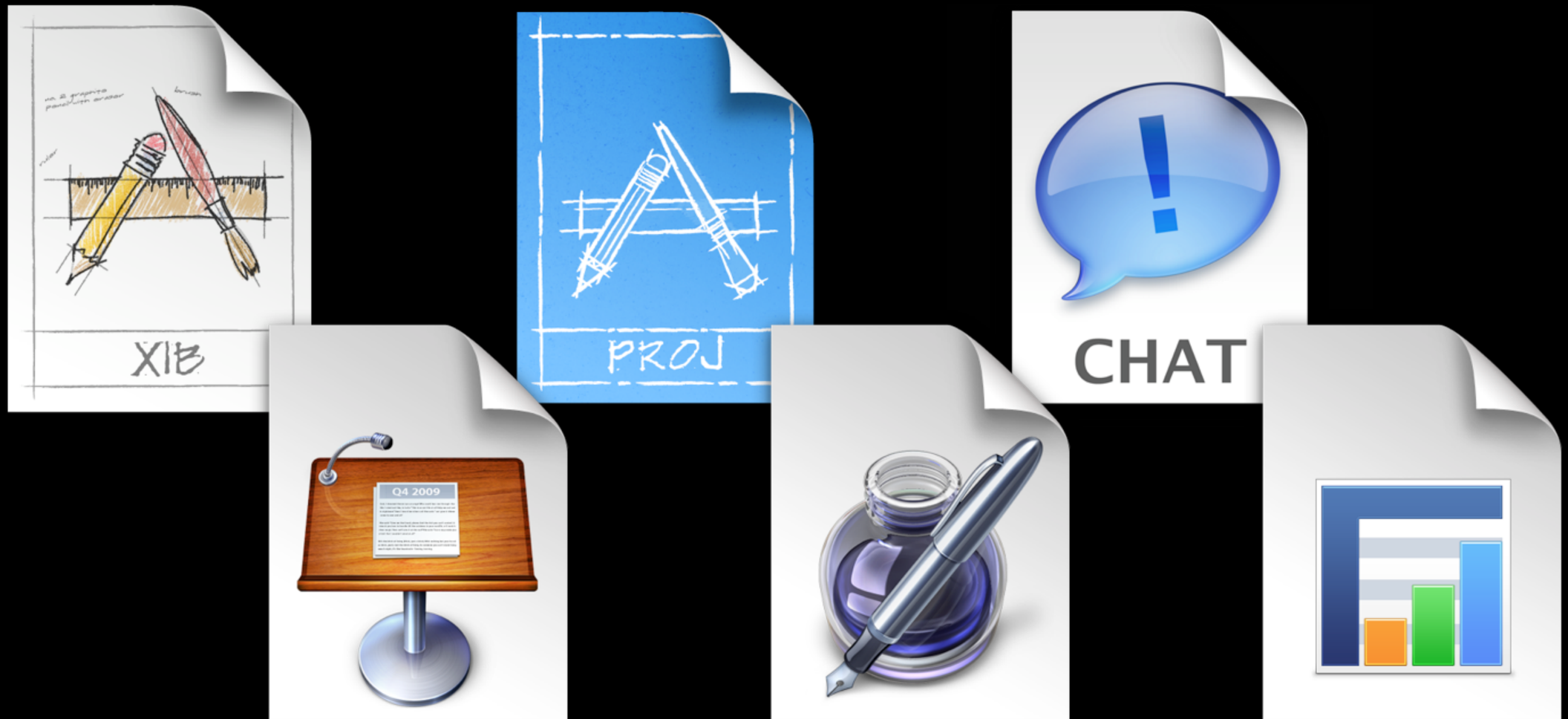

| Lorem Ipsum Dolor | | |
|---|---|---|
| Eiusmod | Mnuit | Dodtos |
| Lorem | ••• | 125 |
| Ipsum | •• | 145 |
| Dolor | ••• | |
| Ahmet | •••• | |



*…tionem consuetudium*
*…m est notare quam;*
*… nunc nobis videntur*
*…ant sollemnes in*
*…n processus dynamicus*
*…utationem*
*consuetudium lectorum Mirum est, etiam processus dynamicus qui sequitur.*

**Lorem ipsum dolor sit amet, ligula suspendisse nulla.**

# Document-Based Applications



# Shoebox Applications

# First Decisions

- Use NSDocument?
- Use Core Data?

# Use NSDocument?

- Yes, if your application is document-based

# Advantages of NSDocument

- Less code
- Consistent behavior
- Correct handling of subtleties
- Integration with other technologies

# More About NSDocument

| | |
|---|---|
| Cocoa Open Lab | OS X Lab C<br>Tuesday, 3:30PM |
| Cocoa Open Lab | OS X Lab C<br>Thursday, 2:00PM |

## Documentation

Document-Based Application Overview
Application Architecture Overview

# Use Core Data?

- What kind of data do you have?
- Core Data manages object graphs
- Core Data can manage part of your data

# Advantages of Core Data

- Object graph management
- Persistence
- Designed for bindings and controllers
- Use your model code on iPhone
- Integration with other technologies
  - Sync
  - Spotlight

# Object Graph Management

- Undo support
- Validation
- Conflict resolution
- Change notification
- Relationship management
- Delete rules

# More About Core Data

| | |
|---|---|
| What's New in Core Data | North Beach<br>Wednesday, 10:30AM |
| Core Data Lab | Snow Leopard Lab<br>Friday, 9:00AM |

## Documentation

Core Data Overview
Core Data Programming Guide

# Where Does Data Go?

- Your application
- User data
- Other application data

# Where Does Your Application Go?

- Wherever the user wants!
- Self-contained and relocatable
- A single .app bundle if possible

# Where Does User Data Go?

- For document-based apps
  - Where ever the user wants!
  - Often in ~/Documents
- For Shoebox apps
  - ~/Library/Your App Identifier
  - Or possibly in ~/Movies, ~/Music or ~/Pictures

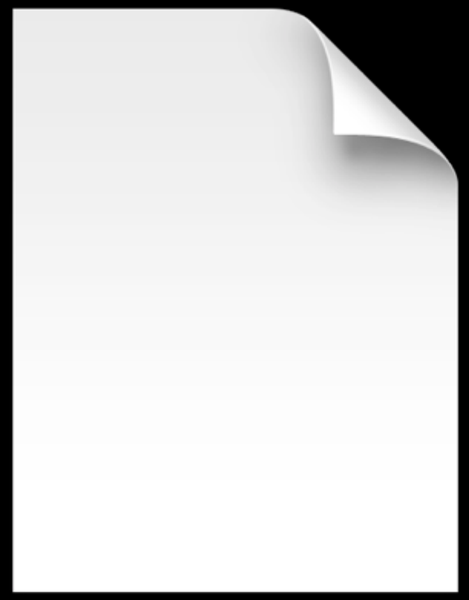
# Where Does Other Data Go?

- Support data should go in Application Support folder
- Preferences should be managed through NSUserDefaults
- Caches should go in the Caches folder
- Temporary data should go in a temp folder

# Finding Standard Folders

- NSSearchPathForDirectoriesInDomains()
  - Applications - NSApplicationDirectory
  - Documents - NSDocumentDirectory
  - Library - NSLibraryDirectory
    - Application Support - NSApplicationSupportDirectory
    - Caches - NSCachesDirectory
- NSTemporaryDirectory()

# Back to the Fun Stuff



NSDocument

Core Data

# Plays Well with Others

# Plays Well with Others

- No application is an island
- Users want to
  - Find things
  - Preview things
  - Share things with others
  - Sync things

# Identifying Your Data—UTIs

- Data comes in different types
- Data types are distinguished using UTIs
- UTIs map to MIME types and path extensions
- Used by document-based apps, Spotlight, and Quick Look

**Documentation**

Uniform Type Identifiers Overview

# Spotlight

- Systemwide searches
- Uses indexed metadata
- Plug-ins do the indexing
- Your application can…
  - Provide importer plug-ins
  - Perform its own searches

# Spotlight Importer Plug-Ins

- Declare the UTI(s) the importer understands
- Importers discover metadata
  - Define a metadata schema
- Prefer standard metadata attributes
  - kMDItemEmailAddresses, kMDItemAuthors, etc.
  - There are lots of standard attributes
- Define custom attributes when necessary

# Separate Files

- Spotlight works on files
- Documents in document-based applications
- What about shoebox applications?
  - Write out separate files
  - One for each record that should be searchable
  - Core Data can help you with this!

# Spotlight Searches in Your Application

- Applications can perform Spotlight searches
- Consider how this might benefit your users
  - Allow the user to explicitly perform searches
  - Automatically find other context relative data on the system
  - Bring relevant stuff directly to the user

# More About Spotlight

| Spotlight for Snow Leopard Lab | Snow Leopard Lab<br>Wednesday, 9:00AM |
| --- | --- |

## Documentation

Spotlight Importer Programming Guide
Spotlight Query Programming Guide

# Quick Look

- Systemwide previews
- Thumbnails
- Plug-ins provide the support
- Your application can…
  - Provide Quick Look plug-ins
  - Use previews and thumbnails

# Quick Look Plug-Ins

- Declare the UTI(s) the plug-in understands
- Implement the callbacks
  - Generate thumbnails
  - Generate previews

# Quick Look Plug-Ins—Callbacks

- Thumbnail callback
  - Provide or generate an image
- Preview callbacks
  - Provide or generate an image
  - Draw previews into a graphics context
  - Convert to a common format (text, HTML, RTF, PDF)

# Using Quick Look in Your Application

- Application can use Quick Look
  - To get thumbnail images
  - To present previews in a standard preview panel
- Consider how this might benefit your users
  - Show users non-native data within your application
  - Bring relevant stuff directly to the user

# More About Quick Look

| | |
|---|---|
| Quick Look on Snow Leopard Lab | Snow Leopard Lab<br>Tuesday, 2:00PM |

## Documentation

Quick Look Programming Guide

# iChat Theater

- Present data via video chat
- Documents or any content

# Supporting iChat Theater

- Basic support for free through Quick Look
- Advanced support through session-based API
  - Present any NSView
  - Use a pixel buffer
  - Use a GL buffer
  - You can provide audio channels as well

# More About iChat Theater

| | |
|---|---|
| iChat Lab | OS X Lab B<br>Wednesday, 2:00PM |

## Documentation

Instant Message Programming Guide

# Sync Services

- Keep copies of data synchronized
  - iPhones and iPods
  - Multiple Macs
  - Other devices

# Using Sync Services

- Define a sync schema for your data
  - What data should be synced
  - How is sync identity defined for your records
- Sync Services API to manage sync sessions

# Using Sync Services with Core Data

- Core Data has direct support
  - Sync schema definition through Core Data model annotations
  - Automatic local storage of your data
  - Tracking changes between sync sessions

# More About Sync Services

| Sync Services Lab | OS X Lab D<br>Tuesday, 3:30PM |
|---|---|

## Documentation

Sync Services Tutorial

Sync Services Programming Guide

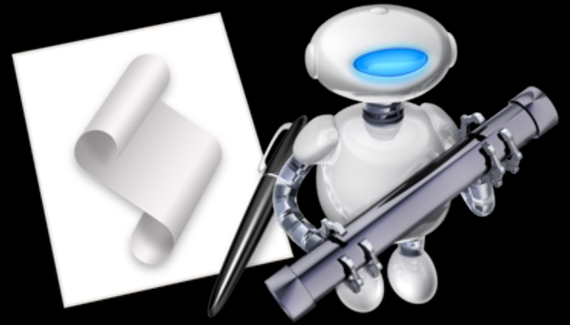
# Plays Well with Others

- Users want to
  - Find things
  - Preview things
  - Share things with others
  - Sync things
- Let them!

# An App for Everyone

# An App for Everyone

- Your potential users
  - Come from many places
  - May have a disability
  - May look to your app as part of the solution to a larger problem



# Universally Useful

- Internationalization
- Accessibility
- Scripting
- Services

# Internationalization

- Support different regions
- Includes
  - Localizability
  - Handling international content

# Localizability

- Mac OS X makes it easy
  - Easy per-region user interface specifications
  - NSBundle and localized resources
  - String table generation and look up

# International Content

- There are differences between regions
  - Language and writing systems
  - Date, time, and number formats
- Use built-in support
  - Unicode and Cocoa text
  - NSDate, NSCalendar, NSDateComponents
  - NSNumberFormatter, NSDateFormatter

# More About Internationalization

| | |
|---|---|
| Building Localized Mac and iPhone Applications | Presidio<br>Friday, 10:30AM |
| Mac and iPhone Internationalization Lab | iPhone Lab D<br>Friday, 3:30PM |

## Documentation

Internationalization Programming Topics

# Accessibility

- Access for those with disabilities
- Lots of support in Cocoa

# Making Your App Accessible

- Support full keyboard navigation
- Don’t override system key equivalents
- Provide alternatives for drag & drop
- Allow canceling and back-tracking in workflows
- Consider in particular
  - Visual disabilities
  - Hearing disabilities
  - Cognitive and motor function disabilities

# Cocoa Accessibility Support

- Associate descriptions with UI elements
- Link-related UI elements
- Custom UI
  - Expose substructure and accessibility info
  - Support relevant attributes and actions
- Use Accessibility Inspector and Voice Over to check your app

# More About Accessibility

| | |
|---|---|
| Enabling Your Mac Applications for Assistive Technologies | North Beach<br>Wednesday, 5:00PM |
| Advanced Accessibility: Best Practices and New APIs in Snow Leopard | Mission<br>Friday, 2:00PM |
| Mac OS X Accessibility Lab | OS X Lab A<br>Thursday, 2:00PM |
| Mac OS X Accessibility Lab | OS X Lab D<br>Friday, 2:00PM |

## Documentation

Accessibility Overview
Accessibility Programming Guidelines for Cocoa

# Scripting

- Give your users flexibility
  - To construct larger workflows
  - To automate repetitive tasks
  - To extend your application

# Making a Scriptable Application

- Start early
- Define a scripting terminology
- Script at the model level
- Provide key-value coding
- Implement handlers for custom commands

# More About Scripting

| | |
|---|---|
| AppleScript and Cocoa Bridge | North Beach<br>Friday, 10:30AM |
| Scriptability Lab | OS X Lab B<br>Wednesday, 12:00PM |
| Applications Scripting and Automation Lab | OS X Lab B<br>Thursday, 5:00PM |
| AppleScript and Cocoa Bridge Lab | OS X Lab C<br>Friday, 2:00PM |

## Documentation

Cocoa Scripting Guide

# Services

- Your features in other contexts
  - Transform data
  - Produce data
  - Trigger your features
- Great for common data types
  - Text
  - Images
  - Files

# Providing Services

- Identify candidate behaviors
  - They should accept and/or return typed data
- Implement a single service provider object
  - Each service is a method on the provider
  - NSPasteboard is used for input and output data
- Declare each service in your application's Info.plist

# More About Services

| | |
|---|---|
| Pasteboards, Services, and Interoperability on Mac | North Beach<br>Thursday, 2:00PM |
| Using Automator and Services to Integrate with Mac OS X | North Beach<br>Friday, 2:00PM |
| Automator and Services Lab | OS X Lab C<br>Friday, 9:00AM |

## Documentation

Services Implementation Guide

# Other Interesting Sessions

| | |
|---|---|
| What's New in Cocoa | Marina<br>Tuesday, 9:00AM |
| State of the Art Cocoa: User Interface Design | Nob Hill<br>Tuesday, 2:00PM |
| Cocoa Tips and Tricks | Nob Hill<br>Thursday, 5:00PM |

# Summary

- Know what's available
- Consider how it applies to your application
- Write really great applications!

## WWDC Session Survey

You can rate your sessions on the WWDC Attendee Site.

Survey forms are located on each session description page at:

developer.apple.com/wwdc/attendee

WWDC09